

(1) Model description : R N N

R06525087 郭漢遜

在這次作業中，我處理D A T A的方式，是將**LABEL**跟**MFCC**按照相同的句子整理對齊好後，然後寫一個函式，可以按照幾個**PHONE**為單位切成一筆**DATA**，同時也就是**Time Step**的長度。

比如輸入50，就會將每一筆句子按照50個**Phone**為一單位切成一個**Array**，而尾巴不足50的部分會補39維的全零**Array**當作**Feature**，因此每一筆句子都變成了所選擇的**Time Step**的倍數；而**Label Data**，在第一步時先將48個**Phone**轉成了39個，原本應該是39維的**one hot**向量，但是有句子不夠要補零的部分，因此多出一維變成了40維：

Input Shape = (Data數 , Time_Step , 39)

Output Shape = (Data數 , Time_Step , 40)

Data數依照裁切長度，如果選擇越短的裁切，總**DATA**的數量也會相應上升，但是句子比較短，這對於R N N的橫向傳遞來說好像是不利的吧？而當時直覺的想法是，感覺一句人話，句子開頭的發音，跟句子中段或結尾的發音通常本來也就沒甚麼關係吧？所以我只要能夠擷取一個發音附近的音也就足夠提升辨識度了吧！只是不太確定到底需要擷取的一段話應該要幾個**Phone**，所以就使用了能夠自由選擇輸入擷取長度的方式來完成作業。

在R N N的嘗試中，使用**KERAS**實現了三層R N N，**units**數量都選擇為256個，而每一層後面都使用**LeakyRelu**為激活函數。會想選**LeakyRelu**的原因，其實不知道到底會不會比**Relu**好，只是單純覺得輸入的資料有正負數，而**LeakyRelu**在尾巴的部分有負數的變化，所以就拿來嘗試了，感覺好了一些，不知道是不是心理作用。

這個三層R N N模型總參數約為30萬個，在選擇將句子以20切成一單位的實驗中(**Input_shape**=(58172, 20, 39))，以**Batch Size** = 128，**validation**比例為0.1的設定下，五次訓練後可以達到0.65的**Validation Accuracy**，表示的確有相當程度的達到辨識能力。

(2) Model description : C N N + R N N

在C N N + R N N的嘗試中，延續了前一個R N N模型，只是在前面多加了兩層的一維捲積，因為不確定要選擇的通道數及捲積大小，所以比照處理影像時候的經驗直覺來說的話，不管怎樣至少要比原有的特徵來的多吧？所以第一層有64通道，第二層為128通道，在**Padding**的方式上選擇了**causal**。而捲積大小則選擇為5，感覺一個音節提取前後5個音節好像已經不錯了！？而模型的總參數為42萬左右。

而實驗的結果應該是錯了！



因為最後驗證準確率在五次訓練後不到0.65左右就停滯了，只是我不知道錯在哪裡而已。

以上兩個Model在每一層後面都使用0.2比例的DropOut層，除了最後一層為SoftMax，每一層都用LeakyRelu激活，Loss Function為categorical crossentropy，優化方式為ADAM。

（ 3 ） Improve my performance

在這次作業實現的過程中，在資料裁切處理完後，前幾天對各種模型的嘗試，始終無法達到有效訓練的成果，不管怎麼使用RNN或CNN或甚麼的，準確率一直停留在0.3左右，實在是個尷尬的數字。

因為如果是有根本性的錯誤，模型準確率在40個結果的亂分類下，準確率應該是2.5%之類極低的數字，但是30%就好像行得通卻又沒有很通的感覺，不斷反覆檢查資料前處理是否有出錯及模型結構會不會太奇怪，都沒有一個結果。

後來靈光一閃想到，在影像前處理時我們會將影像像素從0~255變成0~1之間，如果對這邊的Feature做類似的歸一化處理會不會有效呢！？所以在Input Layer先選擇了Batch Normalization層再開始進入模型的其他層，這樣的調整顯示為非常的有效！一舉從0.3左右提升到了可以過BaseLine的準確度，這幅度簡直是超英趕美！

由於食髓知味的關係，索性就在每一層輸入前都先經過Batch Normalization，但是無法很有效的再提升準確率了！

因此最後各種嘗試的模型主要架構皆為：

Input →

BN層 → 數個卷積層(有或無) → 數個RNN層 → Time Distribute(選擇40個單元)

→ Output

(model compile的方式皆如第二段所述！)

只可惜後來沒有其他更偉大的發現，因此成績只能停留在剛好滑起來滑過BaseLine的地方！

(4) Experimental results and settings

雖然在使用了BN大法之後，準確率從30%左右翻漲了一倍到60%以上，但是其實是不足以通過BaseLine的，必須為這個作業想一些其他的出路。

在LSTM的嘗試中，使用KERAS實現了四層的雙向LSTM，units數量選擇為128 > 128 > 256 > 256個，這樣子總參數會達到三百萬，訓練速度也因此拉長許多，但是結果是好的，驗證準確率的準體水準提升了10%左右，也就是五次訓練可以接近75%，因此將原本的RNN層代換為雙向LSTM之後，再做一些測試及微調，訓練次數大約7次，選出一個稍好的結果，終於剛好BaseLine。

以上結果皆經過多方嘗試，包括改變卷積的通道數或者LSTM的層數及單元數等等，結果皆大同小異，真的很好奇那些排名前面的人使用了甚麼樣的結構跟方法呢！

